REMARKS

The above Amendments and these Remarks are in reply to the conversation with the Examiner on July 18, 2005. Claims 1-41 are currently pending.

Pursuant to the Examiner's requests, the claims submitted herewith have been amended to include identifiers. Also submitted herewith are a Supplementary Declaration and a copy of "Unraveling the Intricacies of Dynamic RAM's", including page 165.

The Commissioner is authorized to charge any underpayment or credit any overpayment to Deposit Account No. 501826 for any matter in connection with this response, including any fee for extension of time, which may be required.

Respectfully submitted,

Date: August 26, 2005

By: _____
Kirk J. DeNiro
Reg. No. 35,854

VIERRA MAGEN MARCUS HARMON & DENIRO LLP
685 Market Street, Suite 540
San Francisco, California 94105-4206
Telephone: (415) 369-9660 x204
Facsimile: (415) 369-9665

DESIGNER'S GUIDE

dynamic RAMs
Part 1

Unraveling the intricacies of dynamic RAMs

Although designers often prefer dynamic RAMs (DRAMs) to their static counterparts, they sometimes shy away from using DRAMs because of the devices' added complexity. This article, part 1 of a 4-part DRAM series, sheds light on some of the complex issues surrounding DRAMs and describes the different DRAM architectures. The succeeding articles will cover memory-system architectures, DRAM controllers, and DRAM-board design.

Steve Gumm and Carl T Dreher,
Texas Instruments

When constructing a memory system, designers most commonly choose the dynamic RAM (DRAM) as the basic building block. The DRAM owes its popularity to the fact that it costs less than the static RAM (SRAM), yet has a higher bit density, an advantage that stems from the DRAM's simple, tiny memory cells. It's true that DRAMs require more housekeeping than SRAMs do: The term "dynamic" implies that the data in the memory cells must continually be accessed or refreshed to ensure that the stored data is valid. Nevertheless, DRAMs' lower cost and smaller size more than make up for any deficiencies.

A designer's first encounter with DRAMs can often be intimidating. The uninitiated engineer must confront such terms as "page-mode access," "static-column mode access," and "nibble-mode access," as well as "refresh rate," and "precharge period." The designer also faces a host of timing parameters, which can be perplexing. Even after you understand these bewildering terms, you must choose from among a vast number of options the DRAM architecture that best suits your needs.

The most common type of DRAM, which is classified as N × 1, stores 1 bit of data in N addressable locations. Each device has a 1-bit input and a 1-bit output data bus. Current single-package bit densities for this family range from 16k bits to 1M bits; 256k bits is the most popular. Other less-common types include the N × 4 and N × 8 DRAMs, which have 4- and 8-bit data buses, respectively. Although this discussion is restricted to N × 1 DRAMs, you can easily apply the information presented here to DRAMs with wider buses.

The difference between SRAMs and DRAMs

Accessing data in a DRAM is different from accessing data in an SRAM. The DRAM has a multiplexed-address arrangement, which conserves package space. To access data, a 256k-byte DRAM requires 18 addresses ($2^{18} = 256k$). To accommodate these addresses on separate pins—along with the normal complement of pins for power, control, and I/O data—would require a 24-pin package. The DRAM has nine address lines (A_0 through A_8) and two strobe lines for accessing data. Essentially, the DRAM accesses a row and a column of an array by internally latching a row address with

Even though they require more housekeeping than SRAMs do, DRAMs' lower cost and component size more than make up for any deficiencies.

the Row Address Strobe (RAS) and a column address with the Column Address Strobe (CAS). Because both addresses are multiplexed onto the same pins, 256k-bit DRAMs come in 16-pin packages and 1M-bit DRAMs come in 18-pin packages.

Besides multiplexed addressing, the most significant differences between SRAMs and DRAMs are their refresh and precharge requirements. A DRAM's memory cell consists of a storage capacitor. When a charge is placed on this capacitor for data storage, the charge must be periodically refreshed; otherwise, the capacitor will discharge, invalidating the stored data. Some of the older 16k-bit DRAMs must be refreshed every 2 msec; newer 1M-bit DRAMs specify a 10-msec refresh rate.

The DRAM's architecture simultaneously refreshes all of the column cells for a selected row every time the row is accessed. Furthermore, some DRAMs reserve the most significant bit (MSB) as a select bit for an internal multiplexer that selects data in two banks of arrays. In these DRAMs, the lower address bits access the memory cells in both banks simultaneously. This arrangement cuts the required number of refresh cycles in half. For example, a 64k-bit DRAM, which has eight row- and column-address lines, requires a refresh cycle for only 128 rows instead of 256. Similarly, a 256k-bit DRAM requires only 256 cycles, and a 1M-bit DRAM requires only 512 cycles.

Another peculiarity of the DRAM is precharge. Performing a memory-read operation from a DRAM cell causes its capacitor to discharge slightly. Therefore, data must be written back into the cell after each read operation. This write-back operation is called "precharge" and is automatically handled by the DRAM. The time it takes to perform this operation is called the precharge period, and you must account for it when you determine the read-cycle time for the DRAM.

An N×1 DRAM has two pins, D and Q, for handling the input and output (I/O) data. In most of these DRAMs, the data-out line (Q) goes to the high-impedance state when inactive, so you can tie it to the data-input line (D) in order to handle bidirectional data. Other DRAMs require external 3-state buffers to handle bidirectional data. In order to conserve package size, N×4 and N×8 DRAMs share a set of bidirectional pins.

Perhaps the most intimidating features of DRAMs are their timing diagrams. Most of the turmoil centers on the sheer number of timing specifications and the combination of abbreviations used to describe them.

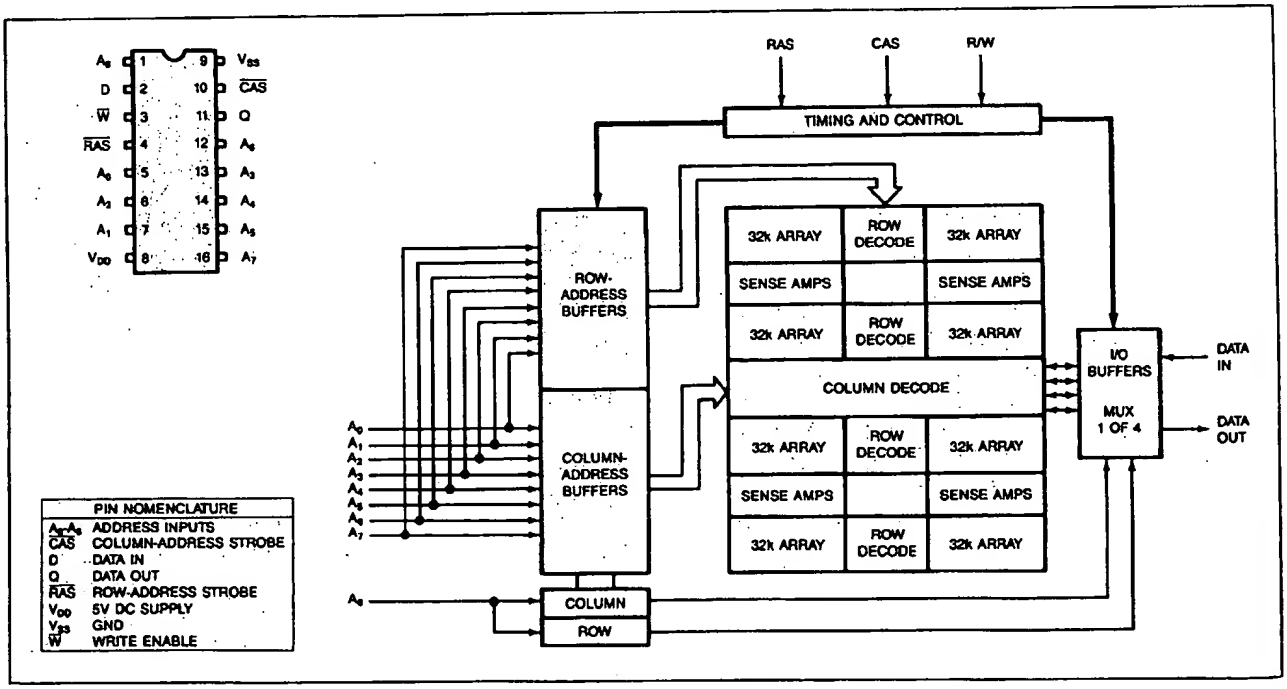


Fig 1—A DRAM's internal architecture determines which of the variety of DRAM modes it can support. The TMS4256-12 DRAM, for example, has a 4-to-1 multiplexer that can access four memory arrays to support nibble-mode access.

TABLE 1—PARAMETER ABBREVIATIONS

PARAMETER	ABBREVIATION
COLUMN OR CAS	C
ROW OR RAS	R
LOW SIGNAL	L
HIGH SIGNAL	H
ACCESS TIMES	a
HOLD TIMES	h
SETUP TIMES	su
PULSE WIDTH	w
READ OPERATION	rd
WRITE OPERATION	W
CYCLE	c
ADDRESSES	A
DATA	D

The abbreviations listed in **Table 1** will help you decipher the timing specifications. For example, the minimum pulse width during which the $\overline{CAS}$ line is low is abbreviated as "tw(CL)." The w indicates pulse width, the C indicates $\overline{CAS}$, and the L means "low." Similarly, the setup time for a row address is abbreviated as "tsu(RA)."

To accommodate a DRAM's timing requirements, the memory system requires a DRAM controller, which generates the control signals and interfaces the DRAM to the system's CPU. Some CPUs, such as the Z80, include a DRAM controller on chip. Most CPUs, however, require an external circuit that functions as the DRAM controller. The controller must coordinate the system access while guaranteeing that the DRAM is refreshed within its specified period. Ideally, the DRAM controller fools the system CPU into believing that it's communicating with an SRAM.

To use a DRAM effectively, you must pay attention to a number of timing parameters. The read cycle is probably the easiest of these parameters to understand, and it also demonstrates many of the DRAM's timing principles. **Fig 1** shows the timing parameters for a 256k-bit DRAM (the TMS4256-12).

Reading and writing

A read cycle begins when the controller issues a row address and drives the $\overline{RAS}$ line low, thus strobing the address into the DRAM. The address must be stable for tsu(RA) seconds (row-address setup time) before the falling edge of the $\overline{RAS}$ line, and it must remain stable for a minimum of th(RA) seconds (row-address hold time) after the falling edge (**Fig 2a**). The $\overline{RAS}$ line must remain low during the entire read cycle. Because the $\overline{CAS}$ line not only strobes in the column address but also enables the output buffer, it should be high when strobing the row address into the DRAM. Keeping the $\overline{CAS}$ line high ensures that the Q output is in a high-impedance state.

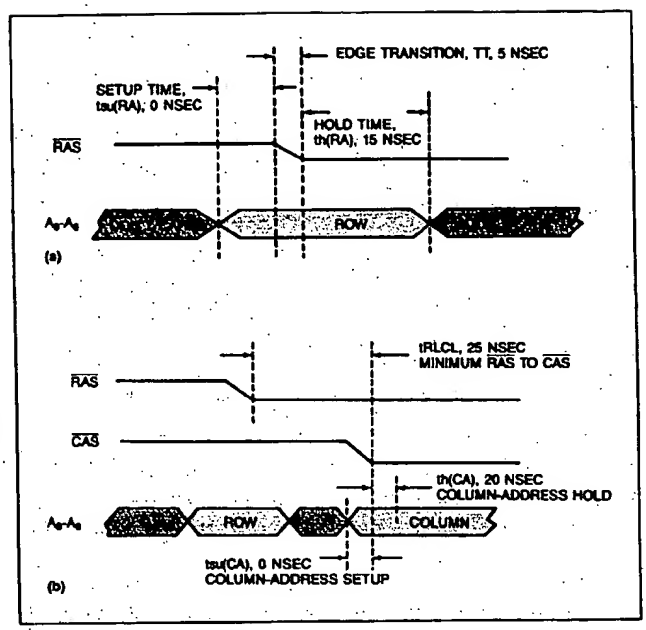


Fig 2—The timing diagrams for a DRAM's read cycle illustrate many of the device's timing constraints. The timing parameters shown here are for the TMS4256-12 DRAM's $\overline{RAS}$ (a) and $\overline{CAS}$ (b) lines, which strobe a row and a column address, respectively, into the DRAM.

Next, the controller must issue a column address to the DRAM. Once the column address is stable for the minimum setup time of tsu(CA) seconds, the controller drives the $\overline{CAS}$ line low to strobe the address into the DRAM. The falling edge of the $\overline{CAS}$ line should not occur any sooner than tRLCL (RAS-low to CAS-low time) seconds after the falling edge of the $\overline{RAS}$ line (**Fig 2b**). Because the $\overline{CAS}$ line also enables the output buffer, the $\overline{WRITE}$ line should be high in order to activate the read cycle when $\overline{CAS}$ is low. After the falling edge of the $\overline{CAS}$ line, there is an access delay of ta(C) seconds before the output buffer is enabled.

To ensure that the output data is valid, you must observe one more critical timing parameter, ta(R). This parameter specifies the minimum time that must elapse after the falling edge of the $\overline{RAS}$ line before the output data is valid. Even though the controller satisfies the requirements for the other delays, such as ta(C), the output data is not valid until ta(R) seconds. For this reason, manufacturers quote ta(R) when specifying a DRAM's speed.

The DRAM's multiplexed-address arrangement conserves package size.

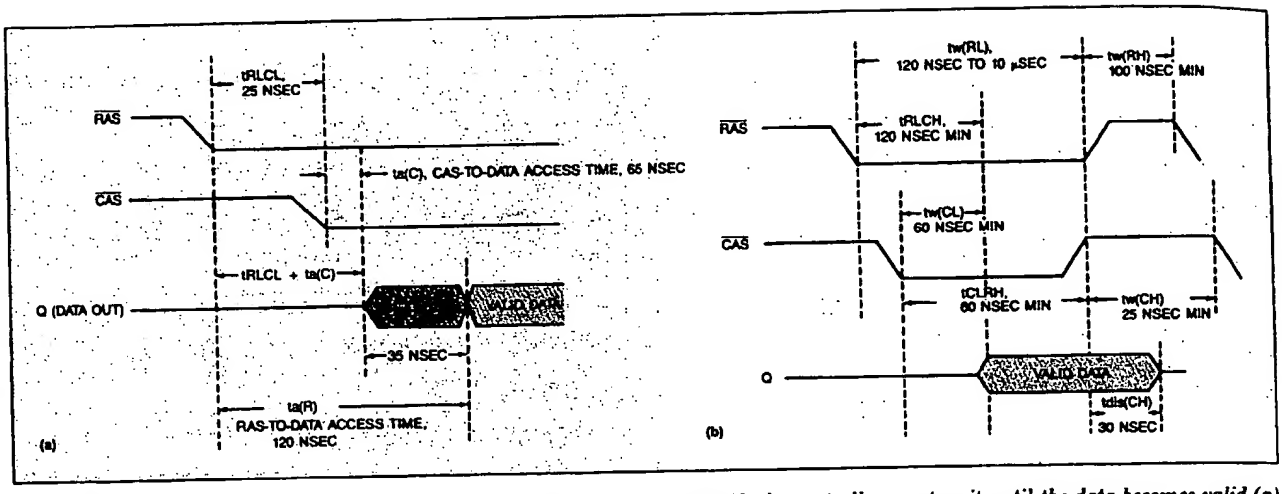


Fig 3—Once the strobe lines load a row and a column address into the DRAM, the controller must wait until the data becomes valid (a) before it can read the data on the Q bus. After reading the data, the controller resets the strobe lines high (b).

To understand the importance of the ta(R) parameter, consider the following example. The TMS4256-12 DRAM specifies a minimum time of 25 nsec for the tRLCL parameter, and it specifies ta(C) as 60 nsec. Therefore, data appears at the output 85 nsec after the falling edge of the $\overline{RAS}$ line. However, the DRAM's ta(R) specification is 120 nsec. Even though data is available at the output 85 nsec after the falling edge of the $\overline{RAS}$ line, it is not valid until 120 nsec after the falling edge—an additional 35 nsec (**Fig 3a**).

To terminate the read operation, the controller must bring the $\overline{RAS}$ and $\overline{CAS}$ lines high. Each line, however, must remain low for a minimum pulse width (tw(RL) and tw(CL), respectively) before returning to the high state. For the TMS4256-12 DRAM, tw(RL) and tw(CL) are specified as 120 nsec and 60 nsec, respectively. In addition, the $\overline{RAS}$ line should not return high until tCLRH seconds (CAS-low to RAS-high time) after the falling edge of the $\overline{CAS}$ line. Similarly, the $\overline{CAS}$ line should not return high until tRLCH seconds (RAS-low to CAS-high time) after the falling edge of the $\overline{RAS}$ line. For the TMS4256-12, tCLRH and tRLCH are 60 nsec and 120 nsec, respectively (**Fig 3b**).

Returning the $\overline{CAS}$ line high disables the DRAM's output buffer. However, the output line is not in a high-impedance state until tdis(CH) seconds (disable time after CAS-high) after the rising edge of the $\overline{CAS}$ line. A typical value for tdis(CH) is 30 nsec.

Driving the $\overline{RAS}$ and $\overline{CAS}$ lines high does not completely terminate the read cycle. The next cycle can't begin until the DRAM automatically writes data back into the previously accessed location during the precharge period. The precharge period occurs while both the strobe lines are held high for a minimum specified pulse width, tw(RH) and tw(CH). For the TMS4256-12 DRAM, tw(RH) is 100 nsec and tw(CH) is 25 nsec. Because tw(RH) is longer, you must observe it when calculating the read-cycle time.

The only restrictions placed on the $\overline{WRITE}$ line during the read cycle are that it must be high for tsu(rd) seconds before the falling edge of the $\overline{CAS}$ line, and that it must remain high for th(CHrd) seconds after the rising edge of the $\overline{CAS}$ line. Because the TMS4256-12 specifies these times as 0 nsec, the $\overline{WRITE}$ line must simply be high during the time that $\overline{CAS}$ is low.

You can now piece together the timing parameters for determining the DRAM's minimum read-cycle time. Because ta(R) is longer than the sum of tRLCL and ta(C) for the TMS4256-12, it dominates this portion of the cycle. The remainder of the cycle consists of the 25-nsec tw(CH), the 100-nsec tw(RH), the 60-nsec tCLRH, and 30-nsec tdis(CH) minimum time periods. Because all of these time periods occur simultaneously, and tw(RH) is the longest period that must be maintained, the read cycle consists of the sum of ta(R) and tw(RH), or 120 + 100 = 220 nsec. Strictly speaking, you should add the maximum time allotted for the $\overline{RAS}$ to make a transition between logic levels; this maximum time is specified as 5 nsec. Adding 5 nsec for each of the cycle's two transitions produces a minimum read-cycle time of 230 nsec (**Fig 4**).

Writing to a DRAM involves most of the same timing parameters that a read operation does. The primary difference between the read and write operations lies

in the timing parameters associated with the $\overline{\text{WRITE}}$ line. Essentially, you can use one of two methods to write to a DRAM—the early-write cycle or the delayed-write cycle. Your choice of write cycle will depend on your application (see **box, "Weigh early- vs delayed-write cycles"**).

The early-write and delayed-write cycles are distinguished by whether the $\overline{\text{WRITE}}$ line is asserted before or after the $\overline{\text{CAS}}$ line is asserted. During an early-write cycle, the controller drives $\overline{\text{WRITE}}$ low prior to driving $\overline{\text{CAS}}$ low. Fundamentally, the falling edge of the $\overline{\text{CAS}}$ line strobes the input data into the DRAM and acts as a reference for the address setup and hold times. During a delayed-write cycle, $\overline{\text{WRITE}}$ strobes the data into the DRAM after the controller asserts the $\overline{\text{CAS}}$ line low. The address setup and hold times are referenced to the falling edge of the $\overline{\text{WRITE}}$ line.

During a delayed-write cycle, $\overline{\text{CAS}}$ is low for a certain time while $\overline{\text{WRITE}}$ is high, so the output drivers are briefly enabled as if the DRAM were executing a read operation. Therefore, unless the memory array has separate input and output buses, the delayed-write operation can cause bus conflicts **(Fig 5)**. In the early-write cycle, the output drivers are not enabled, because the controller asserts the $\overline{\text{WRITE}}$ line low before setting the $\overline{\text{CAS}}$ low. Therefore, you can tie the D input line directly to the Q output line in order to accommodate a bidirectional bus.

The $\overline{\text{WRITE}}$ line must remain low for at least the period specified by tw(W). (For the TMS4256-12, tw(W) is specified as 40 nsec.) During a delayed-write cycle, the $\overline{\text{WRITE}}$ pulse must also remain low for tw(CLW) and tw(RLW) seconds after the falling edges of the $\overline{\text{CAS}}$ and $\overline{\text{RAS}}$ lines, respectively. (For the TMS4256-12, tw(CLW) is 35 nsec, and tw(RLW) is 95 nsec.) Further, the falling edge of the $\overline{\text{WRITE}}$ line must occur at least tsu(WCH) and tsu(WRH) seconds, respectively, before the $\overline{\text{CAS}}$ and $\overline{\text{RAS}}$ lines return to their high states. Both these setup times are specified as 40 nsec for the TMS4256-12 DRAM.

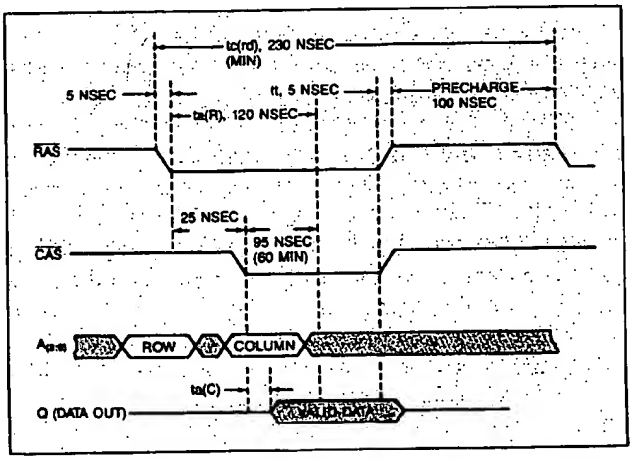


Fig 4—Before the controller can execute two successive reads, it must wait for the DRAM to refresh the previously accessed cell during the precharge period. The minimum read-cycle time for the TMS4256-12 DRAM—230 nsec—includes a 100-nsec minimum precharge period.

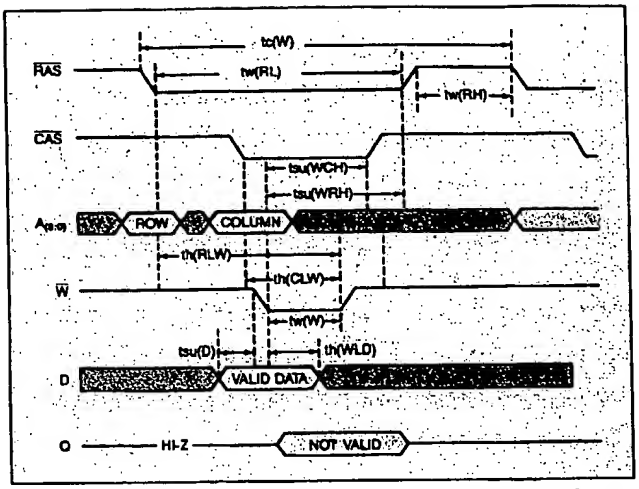


Fig 5—The timing parameters for the delayed-write cycle are appealing for many applications. However, asserting $\overline{\text{CAS}}$ while the $\overline{\text{WRITE}}$ line is high causes the DRAM's output buffers to turn on, which, in turn, causes a potential bus conflict.

In a delayed-write cycle, the data must be stable for at least tsu(D) seconds before the falling edge of the $\overline{\text{WRITE}}$ pulse, and it must remain stable for at least tw(WLD) seconds after the same falling edge. The TMS4256-12 specifies tsu(D) as 0 nsec and tw(WLD) as 35 nsec. Because there is no time limit on the $\overline{\text{WRITE}}$ pulse after the $\overline{\text{CAS}}$ line is driven low, the write operation can be delayed for as long as the refresh period demands.

Don't forget to write

The difference between the early-write cycle and the delayed-write cycle lies mainly in the data setup and hold requirements. In the early-write cycle, the data must be stable for at least tsu(D) seconds before the falling edge of the $\overline{\text{CAS}}$ line. In addition, the data must remain stable for at least tw(CLD) seconds after this falling edge and also for tw(RLD) seconds after the falling edge of the $\overline{\text{RAS}}$ line. The th(CLD) and th(RLD) specifications for the TMS4256-12 are 35 nsec

Besides multiplexed addressing, the most significant differences between SRAMs and DRAMs lie in the devices' refresh and precharge requirements.

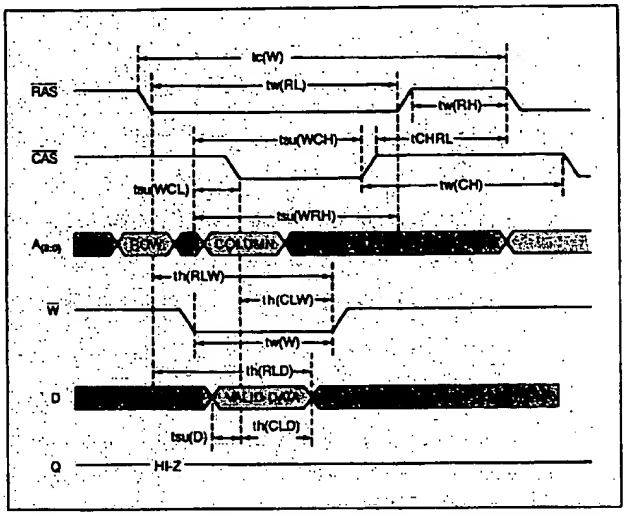


Fig 6—In the early-write cycle, the DRAM controller drives the CAS line low after it sets the WRITE line low. Therefore, the DRAM's Q-output buffers remain in the high-impedance state.

and 95 nsec, respectively. **Fig 6** shows a complete early-write cycle.

Like the read-cycle time, the minimum tw(RL) time exceeds the sum of the other overlapping timing parameters once the controller drives the RAS line low. Therefore, you calculate the minimum write-cycle time (tw(W)), by simply adding tw(RL), tw(RH), and the edge-transition times. The minimum write-cycle time for the TMS4256-12 is 230 nsec (120+100+2×5=230). Because tw(RL) is the dominant write-cycle timing parameter, the minimum write-cycle time is the same for both the early- and the delayed-write cycles.

Before you can read or write to a DRAM, the DRAM controller must execute a power-up sequence that initializes the data in the DRAM. The sequence consists of a short delay followed by a number of write cycles, which charge the memory-cell capacitors. The TMS4256 specifies the delay as 200 μsec plus eight initialization cycles. If you don't include the power-up sequence in the design, a diagnostic test will erroneously detect failures in a perfectly good memory system.

The pause that refreshes

The DRAM controller's highest priority task is to ensure that the DRAM experiences a complete refresh cycle during the specified refresh period. A refresh can occur through a read or write operation or through a specific refresh operation. The TMS4256-12 specifies a refresh period of 4 msec. Because the DRAM's internal architecture has two simultaneously addressed memory banks, the controller needs to refresh only 256 rows during the 4-msec period.

When the controller accesses a row during the refresh cycle, it must adhere to the same timing constraints that it observes for the read and write cycles. Therefore, refreshing all the rows in the TMS4256-12 DRAM takes 256×230 nsec=58.8 μsec. This worst-case refresh-cycle time (58.8 μsec) occupies 1.5% of the allotted 4-msec refresh period. Slower versions of the same DRAM have the same 4-msec refresh-period specification, but occupy a greater bandwidth because of the longer access times.

RAS-only refresh is one of a variety of methods for refreshing DRAMs. In this method, the controller holds the CAS line high while using the RAS line to strobe a row address into the DRAM. The controller remembers which rows have been accessed during a refresh period to ensure that it services all the rows (**Fig 7a**). The DRAM's output drivers are never enabled during a refresh, because the CAS strobe is always high. The minimum time it takes to refresh an entire row is the same as the minimum read-cycle time.

The hidden-refresh method attempts to make refreshing transparent by inserting a refresh cycle when the CPU is decoding an op code after a memory read (**Fig 7b**). For many of today's high-speed μPs, this method is too slow to be practical. The controller initiates the refresh by strobing a row and a column address into the DRAM during a standard memory-read operation. However, instead of going high again at the end of the cycle, the CAS line remains low, latching the output data on the Q bus. Subsequently, the controller toggles the RAS line, causing the DRAM to generate a row address internally. The DRAM contains an internal address generator that sequentially generates a row address on each rising edge of the RAS line when the CAS line is low. During this time, the DRAM ignores the external address bus. Because the DRAM's output drivers are activated throughout the entire cycle, hidden refresh consumes more power than RAS-only refresh.

Automatic CAS-before-RAS refresh combines the best aspects of RAS-only refresh and hidden refresh. In this method, the controller initially asserts the CAS line low before asserting the RAS line low. The DRAM's internal address generator then sequentially generates the row addresses after the controller has toggled the RAS line. Because this method asserts the CAS line low when the RAS line is high, the DRAM's

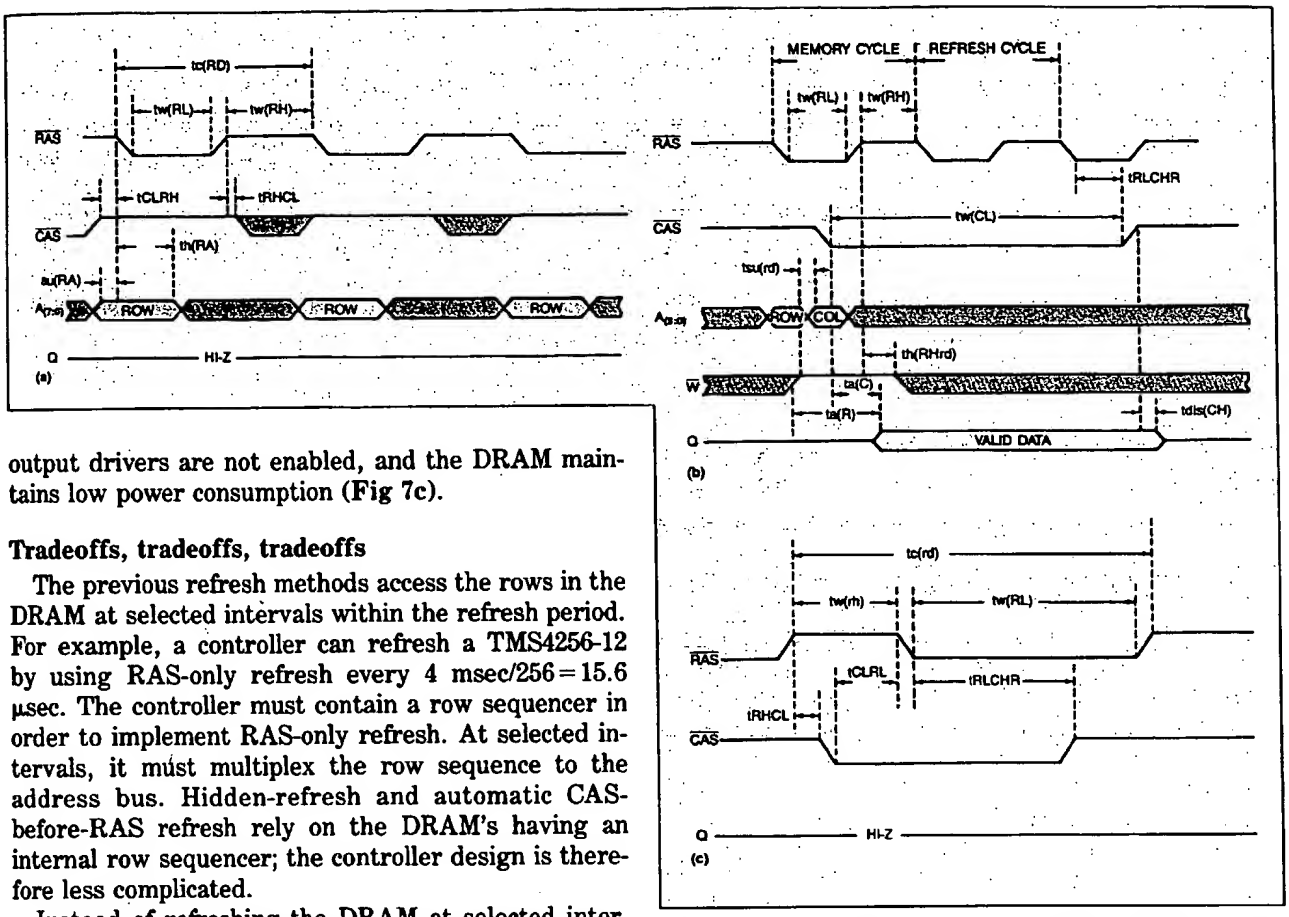


Fig 7—Three common methods for refreshing DRAMs include RAS-only refresh (a), hidden refresh (b), and automatic CAS-before-RAS refresh (b). The controller accesses the DRAM at selected intervals to implement these refresh schemes.

output drivers are not enabled, and the DRAM maintains low power consumption (**Fig 7c**).

Tradeoffs, tradeoffs, tradeoffs

The previous refresh methods access the rows in the DRAM at selected intervals within the refresh period. For example, a controller can refresh a TMS4256-12 by using RAS-only refresh every 4 msec/256 = 15.6 µsec. The controller must contain a row sequencer in order to implement RAS-only refresh. At selected intervals, it must multiplex the row sequence to the address bus. Hidden-refresh and automatic CAS-before-RAS refresh rely on the DRAM's having an internal row sequencer; the controller design is therefore less complicated.

Instead of refreshing the DRAM at selected intervals, burst refresh rejuvenates all the rows in one burst. Although this method furnishes the CPU with a longer period of uninterrupted access time, it produces a very long latency period if the CPU desires access to the DRAM during the refresh burst. Therefore, burst refresh is used only in applications in which the long uninterrupted access time is a must.

Some refresh methods even scrub the data in the DRAM, occasionally removing any soft errors incurred over time. DRAMs are more susceptible to soft errors than are SRAMs. Soft errors are often caused by alpha particles that are emitted by radioactive impurities in a DRAM's package. Although the probability of an individual soft error is small, the cumulative probability of an error increases as the memory size increases. During a scrubbing refresh cycle, the controller reads the accessed data and provides error detection and correction (EDC). The controller then rewrites any corrected data back into the DRAM.

Besides offering various refresh methods, modern DRAMs provide an assortment of modes for accessing data in the storage cells. (**Table 2** summarizes many of the advantages and disadvantages of the available DRAM access modes.) To enhance the access speed, all of these modes rely on the principle that it's not necessary to strobe both the row and the column addresses into the DRAM when you access small portions of consecutive memory space. By eliminating some of the address strobes, these access modes shorten the overall access time and reduce the overall precharge time. Because most DRAMs support only one option, the DRAM controllers for a particular design are often unique, preventing you from mixing DRAMs that have different access modes.

Page-mode memory access establishes a constant row address, while the controller strobes a series of

Performing a read operation from a DRAM cell causes the capacitor to discharge slightly.

column addresses into the DRAM (Fig 8a). The controller strobes both a row and a column address into the DRAM on the first access, but from there on, it strobes only column addresses into the DRAM (by using the $\overline{CAS}$ line) during access periods. The maximum permissible time for tw(RL) (the maximum time for the $\overline{RAS}$ line to remain low) determines the maximum number of columns that are accessible during one page-mode access period.

In page-mode access, the minimum cycle time for strobing a column address into the DRAM is the sum of the minimum tw(CH)P (the $\overline{CAS}$-high pulse width in page mode), the minimum tw(CL) (the $\overline{CAS}$-low pulse width), and the minimum time required for two edge transitions. For the TMS4256-12, this cycle time is 120 nsec. Because the maximum value for tw(RL) is specified as 10 μsec, the controller can address about 83 columns during a single page-mode access period.

In DRAMs that feature an enhanced page-mode option, the internal column-address latch is transparent when the $\overline{CAS}$ line is high. This arrangement gives the DRAM's column decoder direct access to the address when the latch is in its transparent state. An access cycle begins immediately, therefore, when a valid column address appears on the address bus. The transparent latch eliminates the column-address setup-time constraint. The falling edge of the $\overline{CAS}$ line latches the column address, and the rest of the cycle behaves as a standard page-mode cycle (Fig 8b).

Some DRAMs feature a static-column mode for high-speed read and write access. These DRAMs have a 3-state input column-address buffer instead of a col-

Weigh early- vs delayed-write cycle

Most DRAMs offer both early- and delayed-write cycles. Which type of write cycle you choose will depend heavily on your application's timing parameters and system architecture. For example, consider a system design that uses a CPU, such as an 8086 μP, that has a multiplexed address/data bus. The CPU communicates with external memory by placing a destination address on its I/O pins. After a suitable delay, it transfers data to or from memory by using the same I/O pins. An early-write cycle seems well suited for this application because it doesn't need 3-state buffers for eliminating bus conflicts. In either case, you need a DRAM controller to coordinate the action between the two devices.

In the early-write cycle, the CPU initiates a write cycle by placing a row address on its I/O pins. The DRAM controller, which is located between the CPU and the memory, latches the address and begins an early-write cycle by driving the DRAM's $\overline{RAS}$ and $\overline{WRITE}$ lines low. After the CPU places a column address on the bus, the controller latches the address. Before the controller can drive the $\overline{CAS}$ line low, however, it must wait for the CPU to multiplex the data onto the I/O pins. Once the controller determines that the data is stable, it can then drive the $\overline{CAS}$ line low for the minimum tw(CL) seconds. After this procedure, the controller can return the $\overline{CAS}$, $\overline{RAS}$, and $\overline{WRITE}$ lines to their high state.

Now consider using a delayed-write cycle under the same circumstances. Once again, the CPU initiates a write cycle by placing a row address on its I/O pins. The controller latches the address and begins a delayed-write cycle by driving the $\overline{RAS}$ line low. Because the controller holds the $\overline{WRITE}$ line high, it can issue a low on the $\overline{CAS}$ line while it's waiting for the data to stabilize. By driving the $\overline{CAS}$ line low, the controller enables the DRAM's output buffers. Therefore, your system requires additional 3-state buffers in order to prevent bus conflicts. When the controller detects that the data from the CPU is stable, it writes the data into the DRAM by driving the $\overline{WRITE}$ line low for a minimum of tw(WL) seconds during the remainder of the cycle.

In this sample application, the difference between an early-write cycle and a delayed-write cycle is the difference between the DRAM's minimum tw(CL) and tw(WL) specifications. The TMS4256-12 DRAM specifies the minimum tw(CL) time as 60 nsec and the minimum tw(WL) time as 40 nsec. For this application, therefore, the delayed-write cycle is the cycle of choice, even though it will force you to use additional hardware, because it's 20 nsec faster than the early-write cycle. For slower DRAMs, the difference can be even more pronounced.

TABLE 2—COMPARISON OF DRAM MODES

	MODE	ADVANTAGES	DISADVANTAGES
WRITE MODES			
	DELAYED-WRITE	FASTEST ACCESS	REQUIRES 3-STATE BUFFER
REFRESH MODES			
	HIDDEN	DRAM GENERATES ROW ADDRESS INTERNALLY, SIMPLIFYING DRAM CONTROLLER DESIGN; REFRESH IS HIDDEN DURING SLOW CPU CYCLES	Q BUFFER IS ON DURING REFRESH; HIGHER POWER THAN RAS-ONLY; USEFUL ONLY FOR SLOW CPUs
	BURST	AFTER REFRESH BURST, MEMORY IS AVAILABLE FOR MAXIMUM REFRESH PERIOD (4 TO 15 µSEC)	DURING REFRESH BURST, MEMORY IS UNAVAILABLE; REQUIRES SPECIAL DRAM CONTROLLER
ACCESS MODES			
	PAGE	LOW POWER; ALLOWS RANDOM ACCESS WITHIN ANY ROW; DOES NOT REQUIRE SPECIAL DRAMs	NEEDS SOPHISTICATED DRAM CONTROLLER; ACCESS USUALLY SLOWER WHEN CONTROLLER ACCESSES RANDOM ROWS
	STATIC-COLUMN	VERY FAST ACCESS; ALLOWS RANDOM ACCESS WITHIN ANY ROW	HIGH POWER BECAUSE Q BUFFERS ARE ON BETWEEN ACCESSES; REQUIRES 3-STATE BUFFERS; NEEDS SOPHISTICATED DRAM CONTROLLER; SPECIAL DRAMs; ACCESS IS USUALLY SLOWER WHEN CONTROLLER ACCESSES RANDOM ROWS

umn-address latch. When the $\overline{CAS}$ line is high, this input buffer is in the high-impedance state. The controller initiates the first static-column access by asserting the $\overline{RAS}$ line and then the $\overline{CAS}$ line. Holding both lines low throughout the access period, the controller accesses the subsequent columns by altering the column address. This method eliminates the setup, hold, transition, and precharge times associated with toggling the $\overline{CAS}$ line (Fig 8c).

Because DRAMs inherently can't support both page-mode and static-column-mode access, you must choose the mode that best suits your system. Essentially, both modes provide the CPU with a low-cost cache memory by virtue of their fast access to a small block of the system's memory (there are many columns within a single row). One factor in your choice is access-mode speed. Although static-column mode is faster than page mode (because static-column mode eliminates some timing restrictions), the controller must alter the column address quickly enough to realize the speed advantage.

Power consumption is another consideration. Because the output buffers are continually enabled during static-column access, a DRAM that uses static-column mode consumes more power than does a DRAM that uses page-mode access. In either case, when there are frequent row changes, both modes can often be slower than direct read and write cycles because of the extra overhead in the DRAM controller.

The TMS4256 DRAM's architecture offers the use

Ideally, the DRAM controller fools the system CPU into believing that it's communicating with an SRAM.

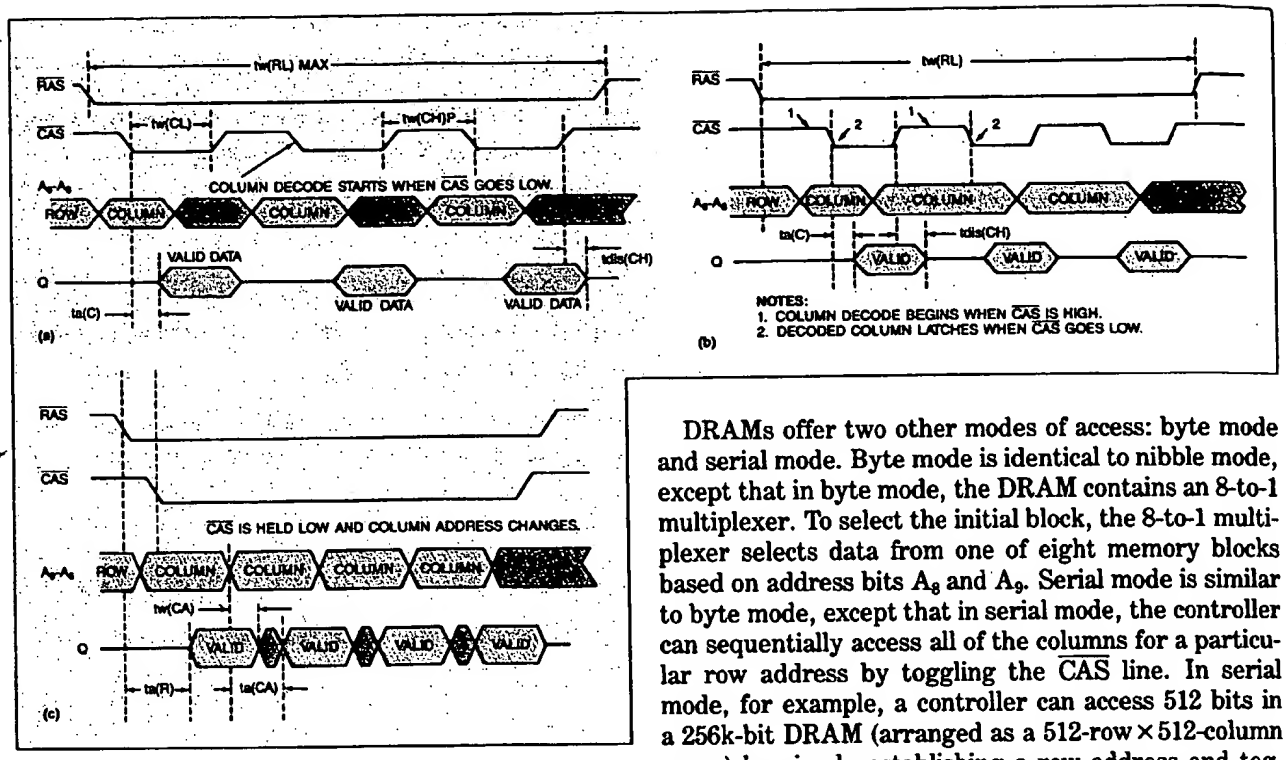


Fig 8—When designing a memory system, you must choose from a number of access-mode options before selecting a DRAM for a particular application. To implement page-mode (a), enhanced page-mode (b), or static-column mode (c) access, for example, you require a controller that's customized to support the mode you choose.

of nibble mode for reading and writing data. As shown in Fig 1, each of the DRAM's dual memory banks are further subdivided into dual 64k-bit blocks. Furthermore, the DRAM contains a 4-to-1 multiplexer, which selectively enables one of the four 64k-bit blocks. Bit A_8 for the row and column addresses determines which block is initially selected. After accessing an initial block, the controller indexes the other three blocks by toggling the $\overline{CAS}$ line three times while maintaining a low on the $\overline{RAS}$ line. Each falling edge of the $\overline{CAS}$ line exercises the multiplexer-select lines circularly through the following sequence:

$$\longrightarrow (0,0) \longrightarrow (0,1) \longrightarrow (1,0) \longrightarrow (1,1) \longrightarrow$$

In this way, the controller can access 4 bits (a nibble) of data with one set of row and column addresses. Typically, the access time for the final 3 bits is 60 nsec/bit.

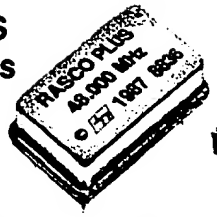
DRAMs offer two other modes of access: byte mode and serial mode. Byte mode is identical to nibble mode, except that in byte mode, the DRAM contains an 8-to-1 multiplexer. To select the initial block, the 8-to-1 multiplexer selects data from one of eight memory blocks based on address bits A_8 and A_9. Serial mode is similar to byte mode, except that in serial mode, the controller can sequentially access all of the columns for a particular row address by toggling the $\overline{CAS}$ line. In serial mode, for example, a controller can access 512 bits in a 256k-bit DRAM (arranged as a 512-row × 512-column array) by simply establishing a row address and toggling the $\overline{CAS}$ line 511 times (after the initial access).

Some access-mode comparisons

In sum, your application will ultimately determine the access mode you choose. All the modes have advantages and disadvantages, and the tradeoffs they present are the familiar ones: performance vs cost and complexity. Page-mode access, for example, requires one row and a separate column address for each location in the DRAM, whereas nibble mode requires only one address and four $\overline{CAS}$ strobes to access data. Therefore, nibble mode is faster than page mode for short, sequential read operations such as fetching 4 bytes of data for a 32-bit CPU via an 8-bit port. However, nibble mode requires a precharge period when the $\overline{RAS}$ line returns high after four successive reads.

Because page mode holds the $\overline{RAS}$ line low throughout the access period, it's well suited to long, uninterrupted data transfers. Although nibble mode may be faster than page mode for long data transfers, when the DRAM controller employs nibble mode it must supply a new set of addresses on every fourth transfer. In addition, page mode can randomly access the columns within a row, whereas nibble-mode access is sequential.

RASCO© PLUS Oscillator Drives CMOS up to 50 MHz



From 1.25 to 50 MHz, drives high speed microprocessors (68020, 68030, 80386, etc.), with ±0.01% stability from 0° C to +70° C. TTL compatible (up to 5 gates), or CMOS compatible (50 pF load). Optional ±0.005% stability, enable/disable. Optional tight symmetry to 50 MHz TTL, or to 20 MHz in CMOS.
John Keilman, 312-451-1000

MSO Surface Mount Data Clocks from 1.25 to 35 MHz

Hermetically sealed ceramic package only 0.560" by 0.360" with 0.160" seated height. "C" lead configuration. Tape-and-reel or anti-static tubes for standard pick-and-place. Vapor phase or wave solder reflowable. ±0.01% stability, CMOS or TTL compatible; 5,000g shock rating. Optional tight symmetry, enable/disable.
John Keilman, 312-451-1000

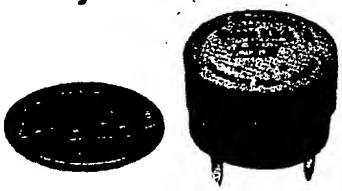
Voltage Controlled Oscillator for Phase Locked Loop

The K1523AA Voltage Controlled Crystal Oscillator allows system designer to phase lock to a reference standard. For LANs, computer shared management systems, communications, A/D interface. 3 MHz to 35 MHz, ±50 ppm per volt sensitivity. ±25 ppm stability. TTL or CMOS compatible.
John Keilman, 312-451-1000

ECL-Compatible Oscillator 40 to 150 MHz

MECL 10KH-based design for improved rise and fall times, duty cycle, noise margin, and power supply rejection. Open emitter output for user selection of load. Complementary outputs available, various pinouts also available.
John Keilman,312-451-1000

Miniature Memory Backup Battery



Rechargeable NiCd batteries use less area than a dime on a pc board, deliver 35 mAh of memory backup power. MMB "C" series batteries are available in 2.4, 3.6, or 4.8V EMF, with seated heights of .280", .450" and .650"; slightly larger "B" series batteries offer 110 mAh capacity. Can be wave soldered, no tie-downs needed.
John Keilman, 312-451-1000

Crystal Controlled MSO

Surface Mount Oscillators

pick & place from anti-static tube or 24mm tape & reel

- 1.25 to 35 MHz Frequency Range
- 0.560" by 0.360" (14,22 by 9,14 mm) footprint
- Compatible with high speed CMOS or TTL logic
- 5,000g Shock Rating
- Custom IC Chip – Fewer Components, Fewer Interconnects
- Optional Enable/Disable With CMOS or TTL

Automatic Placement - Ideal for use with the latest pick-and-place automated assembly equipment. Available with either tape and reel or anti-static tube packaging, MSO oscillators are vapor phase/wave solder reflowable.

Product Features - Available over a frequency range of 1.25 to 35 MHz, with stability of ±0.01% over the temperature range of 0° C to +70° C. Symmetry is 40/60% standard, 45/55% optional. Compatible with either CMOS or TTL logic.

Total Ceramic Package - For better long term stabillity; provides more complete hermetic seal than available with most plastic packages.

5,000g Shock Rating - Uses AT strip crystal with mechanical shock performance greatly superior to that of conventional crystals – made by a proprietary process that avoids the edge faults, fractures, and weaknesses inherent to conventional processes.

Second Sourcing Available

Custom IC Chip - patented custom IC chip replaces hybrid components , allows fewer processing operations and fewer circuit interconnects. Class 100 clean room, completely automatic processing, statistical process control, and 100% computer-automated testing provide uniform high quality and reliability.

Champion Technologies, Inc.

2553 N. Edgington Street, Franklin Park, IL 60131 (312) 451-1000
FAX: 312-451-7585 EasyLink: 62931824 TELEX: 499-0104 CTI UD

CIRCLE NO 73

Static-column mode offers all of the speed advantages of nibble mode and also provides the random-access capability of page mode. Although static-column mode is more difficult to implement, it's rapidly becoming the mode of choice for high-speed memory systems.

Of course, a DRAM's cost, availability, packaging, and alternate sources will also weigh heavily in your choice of a memory unit for a particular application. Remember, however, that once you select a DRAM, your system's access mode is fixed. A page-mode DRAM, for example, can't operate in a system designed for static-column or nibble-mode DRAMs. You must customize the DRAM controller and the system architecture for your chosen access mode. EDN

Authors' biographies

Steve L Gumm is a senior member of the technical staff at Texas Instruments' Information Systems & Services Div (Dallas, TX). He has been with the company for five years. Steve, who earned a BSEE in 1974 from the Tennessee Technological University, has also worked at General Dynamics and the Tennessee Valley Authority. He's currently a member of the IEEE Computer Society. Steve devotes some of his leisure hours to bicycling, scuba diving, and camping.

Carl T Dreher is a technical writer in the ASIC Marketing Div of Texas Instruments (Dallas, TX). He was previously self-employed at Analytic Instruments and is still a partner in that firm. Carl received a BS in electrical engineering and physics from Rose-Hulman Institute of Technology, and MS and PhD degrees from the University of Virginia. He holds three patents. In his spare time he teaches English ritual dancing and plays the concertina.

Article Interest Quotient (Circle One)
High 482 Medium 483 Low 484

This Page is Inserted by IFW Indexing and Scanning Operations and is not part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ BLACK BORDERS
- ☐ IMAGE CUT OFF AT TOP, BOTTOM OR SIDES
- ☐ FADED TEXT OR DRAWING
- ☑ BLURRED OR ILLEGIBLE TEXT OR DRAWING
- ☑ SKEWED/SLANTED IMAGES
- ☐ COLOR OR BLACK AND WHITE PHOTOGRAPHS
- ☐ GRAY SCALE DOCUMENTS
- ☐ LINES OR MARKS ON ORIGINAL DOCUMENT
- ☐ REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY
- ☐ OTHER: _____

IMAGES ARE BEST AVAILABLE COPY.
As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.